AIAA SciTech
4-8 January 2016, San Diego, California, USA
54th AIAA Aerospace Sciences Meeting

Development of a High-Order Space-Time Matrix-Free Adjoint Solver

Marco A. Ceze*, Laslo T. Diosady†, and Scott M. Murman‡

NASA Ames Research Center, Moffett Field, CA, USA

This paper describes the development of a high-order, matrix-free adjoint solver for the Navier-Stokes equations discretized using the discontinuous-Galerkin method in both spatial and temporal directions. The matrix-free framework allows the use of very high orders of approximation accuracy (e.g.: 8^{th}, 16^{th}) at an affordable computational cost. Our final goal with high-order space-time adjoints is to adaptively improve the solution-space based on output error estimates of simulations of three-dimensional, separated flows. This paper is a progress report on that goal: we present implementation details and verification of our matrix-free adjoint solver in three spatial dimensions for steady and unsteady flows. Additionally, we present an extension of a quasi-Newton approach for the adjoint solve that offers considerable speedup. Finally, we draw a plan for future developments.

I. Introduction

The growth in computational power and algorithm development in the past few decades has granted the science and engineering community the ability to simulate flows over complex geometries, thus making Computational Fluid Dynamics (CFD) tools indispensable in analysis and design. Currently, one of the pacing items limiting the utility of CFD for general problems is the prediction of unsteady turbulent flows.[1–3] Reynolds-averaged Navier-Stokes (RANS) methods, which predict a time-invariant mean flowfield, struggle to provide consistent predictions when encountering even mild separation, such as the side-of-body separation at a wing-body junction. NASA's Transformative Tools and Technologies project is developing both numerical methods and physical modeling approaches to improve the prediction of separated flows. A major focus of this effort is efficient methods for resolving the unsteady fluctuations occurring in these flows to provide valuable engineering data of the time-accurate flow field for buffet analysis, vortex shedding, etc. This encompasses unsteady RANS (URANS), large-eddy simulations (LES), Direct Numerical Simulation (DNS), and hybrid LES-RANS approaches such as Detached Eddy Simulations (DES). These unsteady approaches are inherently more expensive than traditional engineering RANS approaches, hence every effort to mitigate this cost must be leveraged. Arguably, the most cost-effective approach to improve the efficiency of unsteady methods is the optimal placement of the spatial and temporal degrees of freedom (DOF) using solution-adaptive methods.

Under the steady-flow assumption, solution-adaptive methods have been demonstrated to be efficient in reducing discretization error by spatially distributing degrees of freedom according to a scalar field – the adaptive indicator – that assesses the importance of different regions of the domain. A robust method for identifying all important regions in the domain is the adjoint-weighted residual which estimates the discretization error in an output of interest (*e.g.:* lift and drag) and the elemental contribution to this error yields an adaptive indicator.[4] Degrees of freedom are added/removed by locally refining/coarsening (h-adaptation) the mesh or by increasing/decreasing (p-adaptation) the scheme's approximation order. The combination and the choice between these options are not trivial tasks and they have been the subjects of many research works.[5–11] Other approaches that remesh[12–14] the domain based on a metric field derived from adjoint-based error estimates and anisotropy measures have also been demonstrated to be efficient in controlling discretization error.

*Postdoctoral Fellow, Oak Ridge Associated Universities, marco.a.ceze@nasa.gov
†Science and Technology Corporation,laslo.diosady@nasa.gov
‡scott.m.murman@nasa.gov

A far less explored research topic is solution adaptation for accurate and efficient simulation of unsteady flows.[14–17] The limited number of research works in space-time adaptivity is partly due to algorithmic challenges to affordably simulate large-scale unsteady flows and to the difficulty of designing and implementing a robust space-time mesh adaptation strategy. Nevertheless, these challenges are worth taking as the two-dimensional results in Ref. 17 show that adjoint-based space-time adaptation can reduce the number of degrees of freedom by one to three orders of magnitude in comparison to uniform refinement, especially when a high level of accuracy is required. More importantly, the authors show that certain heuristic indicators cause the adaptive procedure to converge to the wrong output value.

In steady problems, the cost of an adjoint solution is comparable to the cost of the primal solution with small variations depending on the implementation. The time dependence in unsteady problems reverses the information propagation direction and the adjoint is more conveniently solved backwards in time. At each time step, a residual linearization about the current primal state makes the cost balance between primal and dual solutions depend highly on the implementation. One of the main contributors to the computational cost is forming and storing the residual Jacobian. This cost is twofold, as storing the Jacobian not only requires a large amount of memory but also hampers the computational efficiency due to memory bandwidth usage.

In this work, we avoid these issues by implementing the transpose-Jacobian action on a vector to use the existing matrix-free Krylov infrastructure[18] as the adjoint solver. This work extends our current solver capability that has been demonstrated to achieve high orders of accuracy (larger than 8^{th}-order) on turbulent flows.[19,20] We will use this work as a building block for our goal of output-based space-time *hp*-adaptation which will allow us to perform LES at significantly lower computational cost than currently. In this paper, we will concentrate on the adjoint solver and the adaptation mechanics will be covered in future papers. This paper is organized as follows. In Section II, we describe the discretization followed by the primal solution strategy presented in Section III. Section IV presents the derivation of the adjoint equation and Section V shows results followed by conclusions in Section VI.

II. Discretization

For completeness, we briefly present the discretization of the flow equations. More details are found in Ref. 18. The compressible Navier-Stokes equations are written in conservative form as:

$$\boldsymbol{q}_{,t} + \nabla \cdot (\boldsymbol{f}^I - \boldsymbol{f}^V) = \boldsymbol{g}, \tag{1}$$

where $(\cdot)_{,t}$ denotes partial differentiation with respect to time. Here, $\boldsymbol{g}$ denotes a source term which is zero for the cases in this paper. The conservative state vector is

$$\boldsymbol{q} = \begin{bmatrix} \rho \\ \rho \boldsymbol{u} \\ \rho E \end{bmatrix}, \tag{2}$$

where ρ is the density, $\boldsymbol{u}$ is the velocity vector, and E the total energy. The inviscid and viscous fluxes are given, respectively, by:

$$\boldsymbol{f}^I = \begin{bmatrix} \rho \boldsymbol{u} \\ \rho \boldsymbol{u}\boldsymbol{u}^T + p\boldsymbol{I} \\ \rho \boldsymbol{u} H \end{bmatrix}, \qquad \text{and} \qquad \boldsymbol{f}^V = \begin{bmatrix} 0 \\ \boldsymbol{\tau} \\ \boldsymbol{\tau} \cdot \boldsymbol{u} - \kappa_T \nabla T \end{bmatrix}, \tag{3}$$

where p is the static pressure, $H = E + \frac{p}{\rho}$ is the total enthalpy, $\boldsymbol{\tau}$ the viscous stress tensor, κ_T is the thermal conductivity, $T = p/\rho R$ is the temperature, and R is the gas constant. The pressure is given by:

$$p = (\gamma - 1)\left(\rho E - \tfrac{1}{2}\rho \boldsymbol{u}^2\right), \tag{4}$$

where γ is the specific heat ratio. The viscous stress tensor, $\boldsymbol{\tau}$, is given by:

$$\tau = \mu\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right) - \lambda(\nabla \cdot \boldsymbol{u})\boldsymbol{I}, \tag{5}$$

where μ is the viscosity, and $\lambda = \frac{2}{3}\mu$ is the bulk viscosity.

Applying a change of variables $\boldsymbol{q} = \boldsymbol{q}(\mathbf{v})$, where $\mathbf{v}$ are the entropy variables:

$$\mathbf{v} = \begin{bmatrix} -\frac{s}{\gamma-1} + \frac{\gamma+1}{\gamma-1} - \frac{\rho E}{p} \\ \frac{\rho \boldsymbol{u}}{p} \\ -\frac{\rho}{p} \end{bmatrix}, \tag{6}$$

we rewrite the Navier-Stokes equations as:

$$\boldsymbol{A}_0 \mathbf{v}_{,t} + \bar{\boldsymbol{A}} \nabla \mathbf{v} - \nabla \cdot (\bar{\bar{\boldsymbol{K}}} \nabla \mathbf{v}) \quad = \quad \boldsymbol{g}, \tag{7}$$

with symmetric $\boldsymbol{A}_0 = \boldsymbol{q}_{,\mathbf{v}}$, $\bar{\boldsymbol{A}} = \boldsymbol{f}^I_{,\boldsymbol{q}} \boldsymbol{A}_0 = \boldsymbol{f}^I_{,\mathbf{v}}$ and $\bar{\bar{\boldsymbol{K}}} = \boldsymbol{f}^V_{,\nabla \boldsymbol{q}} \boldsymbol{A}_0 = \boldsymbol{f}^V_{,\nabla \mathbf{v}}$.[21]

We proceed to discretize (7) as follows. The domain, Ω, is partitioned into non-overlapping elements, κ, while the time is partitioned into intervals (time-slabs), $I^n = [t^n, t^{n+1}]$. Define $\mathcal{B}_h = \left\{ \boldsymbol{w}, \boldsymbol{w}|_\kappa \in [\mathcal{P}(\kappa \times I)]^5 \right\}$, the space-time finite-element space consisting of piece-wise polynomial functions in both space and time on each element. We seek a solution $\mathbf{v} \in \mathcal{B}_h$ such that the weak form:

$$\begin{aligned} \boldsymbol{r}^n(\boldsymbol{w}, \mathbf{v}) \quad = \quad & \sum_\kappa \Bigg\{ \int_{I^n} \int_\kappa - \Big(\boldsymbol{w}_{,t} \cdot \boldsymbol{q} + \nabla \boldsymbol{w} \cdot (\boldsymbol{f}^I - \boldsymbol{f}^V) + \boldsymbol{w} \boldsymbol{g} \Big) \\ & + \int_{I^n} \int_{\partial \kappa} \boldsymbol{w} \cdot (\widehat{\boldsymbol{f}^I \cdot \boldsymbol{n}} - \widehat{\boldsymbol{f}^V \cdot \boldsymbol{n}}) \\ & + \int_\kappa \boldsymbol{w}(t^{n+1}_-) \cdot \boldsymbol{q}(t^{n+1}_-) \Bigg\} = \sum_\kappa \int_\kappa \boldsymbol{w}(t^n_+) \cdot \boldsymbol{q}(t^n_-), \end{aligned} \tag{8}$$

is satisfied for all $\boldsymbol{w} \in \mathcal{B}_h$. Here $\widehat{\boldsymbol{f}^I \cdot \boldsymbol{n}}$ and $\widehat{\boldsymbol{f}^V \cdot \boldsymbol{n}}$ denote numerical flux functions approximating the inviscid and viscous fluxes, respectively, while $\boldsymbol{n}$ is the outward pointing normal vector. In this work, the inviscid flux is discretized using the method of Ismail and Roe,[22] while the viscous flux is an interior penalty method. We discretize the temporal direction in slabs, each of which having an Eqn. 8 to be solved. The slabs are coupled via the temporal flux from the previous slab (term involving $\boldsymbol{q}(t^n_-)$).

The space $\mathcal{B}_h$ is represented using a tensor-product basis such that, on each element, $\mathbf{v}$ is given by the product of Lagrange polynomials:

$$\mathbf{v}(\mathbf{x}(\xi), t(\tau)) = \mathbf{V}_{ijkl} \Phi_{ijkl} \qquad \text{with} \qquad \Phi_{ijkl} = \phi_i(\xi_1) \phi_j(\xi_2) \phi_k(\xi_3) \phi_l(\tau), \tag{9}$$

where $\mathbf{x}(\xi)$ defines a mapping from the reference cube, $\xi \subset [-1, 1]^3$, to physical space, while $t(\tau)$ is the mapping from the reference interval $[-1, 1]$ to the time interval $[t^n, t^{n+1}]$. ϕ_i are one-dimensional Lagrange basis functions defined at Gauss-Legendre (GL) points, while $\mathbf{V}_{ijkl}$ are the corresponding nodal values of the entropy variables. The integrals in (8) are evaluated using numerical quadrature. For example:

$$\begin{aligned} & \frac{2}{\Delta t} \int_{I^n} \int_\kappa - \Big(\boldsymbol{w}_{,t} \cdot \boldsymbol{q} + \nabla \boldsymbol{w} \cdot (\boldsymbol{f}^I - \boldsymbol{f}^V)) \Big) \\ \simeq \quad & \Big\{ - \Big(\tau_{,t} \boldsymbol{w}_{,\tau} \cdot \boldsymbol{q} + \nabla_\xi \boldsymbol{w} \cdot (\tilde{\boldsymbol{f}}^I - \tilde{\boldsymbol{f}}^V)) \Big) |J| \Big\}_{\xi_p \xi_q \xi_r \tau_s} w_p w_q w_r w_s, \end{aligned} \tag{10}$$

where $\xi_p, \xi_q, \xi_r, \tau_s$ are one-dimensional GL quadrature points, and w_p, w_q, w_r and w_s are the associated quadrature weights. $|J|$ denotes the Jacobian of the mapping from element reference space to physical space, ∇_ξ denotes differentiation with respect to the reference coordinate ξ, while $\tilde{\boldsymbol{f}}^I = \xi_{,\mathbf{x}} \boldsymbol{f}^I$ and $\tilde{\boldsymbol{f}}^V = \xi_{,\mathbf{x}} \boldsymbol{f}^V$ are the fluxes mapped to the local element coordinate system. In this work we use a quadrature rule with twice as many quadrature points as nodal points in order to reduce the quadrature error (we ensure exact integration of cubic nonlinearities) thereby improving the nonlinear stability of our scheme.[23,24] Later in the text, we will refer to this quadrature as a "de-aliased" quadrature.

The remaining integrals appearing in (8) are evaluated in a similar manner, which may be described as a sequence of three steps:

1. Evaluate the state ($\mathbf{v}$) and gradient ($\nabla_\xi \mathbf{v}$) at the quadrature points.

2. Evaluate the source ($\boldsymbol{g}$) and fluxes ($\tilde{\boldsymbol{f}}^I$ and $\tilde{\boldsymbol{f}}^V$) at the quadrature points.

3. Multiply the source and fluxes with the basis functions ($\boldsymbol{w}$) or gradients ($\nabla_\xi \boldsymbol{w}$).

A key requirement for efficiency at high polynomial order is the evaluation of the first and third steps using the sum-factorization approach,[24,25] which allows the multiplication of the basis functions to be performed as a sequence of one-dimensional operations. This results in a residual evaluation cost which scales as $O(N^{d+1})$ for each space-time element where N is the solution order and d is the number of spatial-temporal dimensions (for unsteady 3D simulations $d = 4$). For a fixed number of spatial-temporal degrees of freedom (i.e. fixed $DOF = N^d$) the residual evaluation scales linearly with the solution order. However for moderate solution orders, $N = 4 - 16$, the increased operation count with solution order may be offset by the use of optimized numerical kernels[24] via SIMD vectorization. As vector length increases in future processor architectures, the aforementioned solution order range can further increase.

III. Primal Solution Strategy

Given the choice of basis functions and quadrature rule, Equation (8) represents a globally coupled system of nonlinear equations which need to be solved for each time-slab. For large three-dimensional simulations the cost of storing the linearization for a single step of an implicit scheme may be prohibitively expensive. Using a space-time formulation, this storage cost is further scaled by the order of the basis used in the temporal direction. In general, all degrees of freedom within an element are coupled, leading to a storage cost which scales as $O(N^d)$ for a fixed number of degrees of freedom. We overcome the memory requirement limitation by using a matrix-free Newton-Krylov method.

A preconditioned GMRES method is used as Krylov solver, which does not require the explicit storage of the Jacobian matrix,[26] instead, it requires only the application of the linearization to each search direction, i.e. we need to compute the linearized residual in the search direction. In this work, the linearized residual is computed directly. As with the residual evaluation, the terms in the evaluation of the linearized residual in a search direction, $\boldsymbol{s}$, are computed as a sequence of three steps:

1. Evaluate the state ($\mathbf{v}$), gradient ($\nabla_\xi \mathbf{v}$), linearized state ($\boldsymbol{s}$) and linearized gradient ($\nabla_\xi \boldsymbol{s}$) at the quadrature points.

2. Evaluate the linearized source $\boldsymbol{g}^{\text{Lin}} = \frac{\partial \boldsymbol{g}}{\partial \mathbf{v}} \boldsymbol{s} + \frac{\partial \boldsymbol{g}}{\partial \nabla_\xi \mathbf{v}} \nabla_\xi \boldsymbol{s}$ and linearized fluxes ($\tilde{\boldsymbol{f}}^{\text{Lin}} = \frac{\partial \tilde{\boldsymbol{f}}}{\partial \mathbf{v}} \boldsymbol{s} + \frac{\partial \tilde{\boldsymbol{f}}}{\partial \nabla_\xi \mathbf{v}} \nabla_\xi \boldsymbol{s}$) at the quadrature points.

3. Multiply of the linearized source and fluxes with the basis functions ($\boldsymbol{w}$) or gradients ($\nabla_\xi \boldsymbol{w}$).

This approach is more expensive than the finite-difference approach often used in Jacobian-free method,[27] however it is insensitive to a step-size parameter, provides the exact linearization, and it is used to compute the solution of adjoint (dual) problems. Note that steps 1 and 3 are transpose of each other, hence, only the flux Jacobians (step 2) need to be transposed for the adjoint residual operator.

The sum factorization approach is used in the application of steps 1 and 3, such that the cost of applying the linearization scales as $O(N)$ for a fixed number of space-time degrees of freedom. We note that with increasing solution order this is more efficient than storing the linearization and computing the linearized residual as a matrix-vector product whose cost scales as $O(N^d)$ (even if there was no cost associated with forming the linearization).

Furthermore, this approach allows us to use different quadrature rules for computing the linearized and nonlinear residuals. In particular, we often use a lower order, or collocated, quadrature rule in the evaluation of the linearized residuals of our Newton-Krylov scheme. The nonlinear residual, however, is evaluated with the de-aliased quadrature metioned in Section II. The use of a collocated quadrature for the linearized residual introduces a linearization error hence our Newton method may be viewed as inexact. The preconditioner used for the primal solve is a matrix-free extension of the diagonalized Alternating-Direction-Implicit (ADI) method applied to our space-time discontinuous Galerkin discretization.[28]

In certain conditions, Newton's method can converge to a local minimum of the residual norm. We address this problem using a diagonal regularization approach analogous to pseudo-transient continuation for steady problems.[29]

IV. Discrete Adjoint Equation

Consider an output $\mathcal{J}(\boldsymbol{\alpha}, \mathcal{V})$, where the state $\mathcal{V}$ satisfies the discrete residual equations $\mathcal{R}(\boldsymbol{\alpha}, \mathcal{V}) = 0$ for a parameter set $\boldsymbol{\alpha}$. We seek variations of the output that are tangent to the manifold $\mathcal{R}(\boldsymbol{\alpha}, \mathcal{V}) = 0$. Assuming the output functional is at least C^1-continuous on $\mathcal{V}$ and $\boldsymbol{\alpha}$ and the residual equations admit a single solution for a given $\boldsymbol{\alpha}$, output variations must be balanced by corresponding residual variations. We can relate these variations by forming the output Lagrangian,

$$\mathcal{L}(\boldsymbol{\alpha}, \mathcal{V}, \boldsymbol{\Psi}) = \mathcal{J}(\boldsymbol{\alpha}, \mathcal{V}) + \boldsymbol{\Psi}^T \mathcal{R}(\boldsymbol{\alpha}, \mathcal{V}) \tag{11}$$

and force its stationarity (or tangency to $\mathcal{R}(\boldsymbol{\alpha}, \mathcal{V}) = 0$) for arbitrary $\delta\mathcal{V}$ and $\delta\boldsymbol{\alpha}$:

$$\delta\mathcal{L}((\boldsymbol{\alpha}), \mathcal{V}, \boldsymbol{\Psi}) = \underbrace{\left(\left.\frac{\partial \mathcal{J}}{\partial \mathcal{V}}^T\right|_{\boldsymbol{\alpha}} + \boldsymbol{\Psi}^T \left.\frac{\partial \mathcal{R}}{\partial \mathcal{V}}\right|_{\boldsymbol{\alpha}} \right)}_{\text{adjoint equation}} \delta\mathcal{V} + \underbrace{\left(\left.\frac{\partial \mathcal{J}}{\partial \boldsymbol{\alpha}}^T\right|_{\mathcal{V}} + \boldsymbol{\Psi}^T \left.\frac{\partial \mathcal{R}}{\partial \boldsymbol{\alpha}}\right|_{\mathcal{V}} \right)}_{\text{sensitivity equation}} \delta\boldsymbol{\alpha} = 0. \tag{12}$$

We solve the adjoint equation for $\boldsymbol{\Psi}$ and use its value to compute the output sensitivity with respect to $\boldsymbol{\alpha}$ via the sensitivity equation. A generalization of the sensitivity equation can be derived in variation form to yield adjoint-weighted error estimates.[4] An integral part of solving the adjoint equation is the transposed action of the residual Jacobian onto a search direction. We perform this operation in a matrix-free manner to prevent the computational cost from scaling super-linearly with solution order.

IV.A. Matrix-free Adjoint Operator

As with the primal solve, we use a GMRES method, which does not require the explicit storage of the Jacobian matrix – only the application of the adjoint operator to each search direction. As with the primal solve, the adjoint residual is computed algebraically exact – apart from possible quadrature differences. The terms in the evaluation of the adjoint residual in a search direction, $\boldsymbol{a}$, are again computed in a sequence of three steps:

1. Evaluate the state ($\mathbf{v}$), gradient ($\nabla_\xi \mathbf{v}$), adjoint state ($\boldsymbol{a}$) and adjoint gradient ($\nabla_\xi \boldsymbol{a}$) at the quadrature points.

2. Evaluate the adjoint source ($\boldsymbol{g}^{\text{Adj}} = \frac{\partial \boldsymbol{g}}{\partial \mathbf{v}}^T \boldsymbol{a} + \frac{\partial \tilde{\boldsymbol{f}}}{\partial \mathbf{v}}^T \nabla_\xi \boldsymbol{a}$) and fluxes ($\tilde{\boldsymbol{f}}^{\text{Adj}} = \frac{\partial \boldsymbol{g}}{\partial \nabla_\xi \mathbf{v}}^T \boldsymbol{a} + \frac{\partial \tilde{\boldsymbol{f}}}{\partial \nabla_\xi \mathbf{v}}^T \nabla_\xi \boldsymbol{a}$) at the quadrature points.

3. Multiply the adjoint sources and fluxes with the basis functions ($\boldsymbol{w}$) or gradients ($\nabla_\xi \boldsymbol{w}$).

We note that steps 1 and 3 are identical to those for the primal problem, while step 2 involves simply the transpose of the operations computed in the linearized residual.

The Jacobians of the fluxes at the quadrature points from Section III are not explicitly constructed in our implementation. Instead, we express the Fréchet derivative as a sequence of algebraic operations and the implementation of their transpose follows an approach similar to the reverse-mode automatic differentiation.

Suppose we compute $\boldsymbol{g} = \boldsymbol{g}(\mathbf{v}) : \mathbb{R}^2 \mapsto \mathbb{R}^2$ using the sequence of scalar steps:

$$\begin{aligned} c &= c(\mathbf{v}) && (13)\\ d &= d(\mathbf{v}, c) && (14)\\ g_1 &= g_1(c, d) && (15)\\ g_2 &= g_2(c, d) && (16) \end{aligned}$$

Then the linearized residual, $\boldsymbol{g}^{\text{lin}} = \boldsymbol{g}^{\text{lin}}(\mathbf{v}, \boldsymbol{s}) : \mathbb{R}^2 \times \mathbb{R}^2 \mapsto \mathbb{R}^2$, in a search direction $\boldsymbol{s}$ is computed using the chain rule:

$$\begin{aligned} c^{\text{lin}} &= \tfrac{\partial c}{\partial v_1} s_1 + \tfrac{\partial c}{\partial v_2} s_2 \\ d^{\text{lin}} &= \tfrac{\partial d}{\partial v_1} s_1 + \tfrac{\partial d}{\partial v_2} s_2 + \tfrac{\partial d}{\partial c} c^{\text{lin}} \\ g_1^{\text{lin}} &= \tfrac{\partial g_1}{\partial c} c^{\text{lin}} + \tfrac{\partial g_1}{\partial d} d^{\text{lin}} \\ g_2^{\text{lin}} &= \tfrac{\partial g_2}{\partial c} c^{\text{lin}} + \tfrac{\partial g_2}{\partial d} d^{\text{lin}} \end{aligned} \tag{17}$$

The adjoint residual, $\boldsymbol{g}^{\text{adj}} = \boldsymbol{g}^{\text{adj}}(\mathbf{v}, \boldsymbol{z}) : \mathbb{R}^2 \times \mathbb{R}^2 \mapsto \mathbb{R}^2$, in a search direction $\boldsymbol{z}$ is simply the reverse operations swaping inputs with outputs:

$$
\begin{aligned}
d^{\text{adj}} &= \frac{\partial g_2}{\partial d} z_2 \\
c^{\text{adj}} &= \frac{\partial g_2}{\partial c} z_2 \\
d^{\text{adj}} &+= \frac{\partial g_1}{\partial d} z_1 \\
c^{\text{adj}} &+= \frac{\partial g_1}{\partial c} z_1 \\
c^{\text{adj}} &+= \frac{\partial d}{\partial c} d^{\text{adj}} \\
g_2^{\text{adj}} &= \frac{\partial d}{\partial v_2} d^{\text{adj}} \\
g_1^{\text{adj}} &= \frac{\partial d}{\partial v_1} d^{\text{adj}} \\
g_2^{\text{adj}} &+= \frac{\partial c}{\partial v_2} c^{\text{adj}} \\
g_1^{\text{adj}} &+= \frac{\partial c}{\partial v_1} c^{\text{adj}}.
\end{aligned}
\tag{18}
$$

Note that the operations in (18) are equivalent to $\boldsymbol{g}^{\text{adj}} = \frac{\partial \boldsymbol{g}^{\text{lin}}}{\partial \mathbf{v}}^T \boldsymbol{z}$.

Through optimized kernels, we can further improve the linear scaling provided by the sum factorization approach and achieve approximately constant cost for the linearized residual evaluation.[18] Figure 1 compares the forward and transpose Fréchet derivatives for various approximation orders. Note that the CPU time spent per degree-of-freedom increases slowly in the range of 2^{nd} to 10^{th} approximation order in space and time. The transpose action of residual Jacobian is slightly more expensive than the forward application. We expect this offset to be reduced or eliminated by further optimizing the transpose operations.

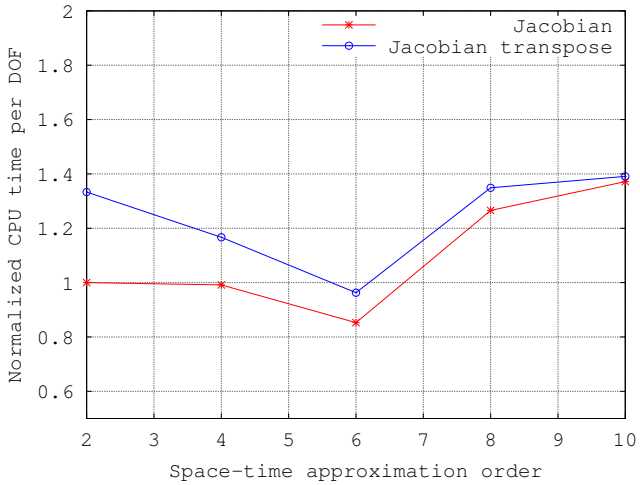


Figure 1: Computational cost per degree of freedom of the forward and transpose Fréchet derivatives – Computed on one processor core of Intel Xeon E5-2680v2.

The preconditioner for the adjoint system uses a diagonalized Alternating Direction Implicit (ADI) algorithm that we have previously developed for the primal problem problem.[28] The preconditioner for the primal problem involves the solution of one-dimensional scalar advection-diffusion systems along each reference coordinate direction (three spatial and one temporal) of every element, with a transformation to/from characteristics variables prior to each solve. Similarly, the preconditioner for the adjoint problem involves the solution of a one-dimensional scalar advection-diffusion adjoint problems in each coordinate direction. For

the adjoint problem, we perform the directional solves in the opposite direction (i.e. τ,ξ_3,ξ_2,ξ_1 as opposed to ξ_1,ξ_2,ξ_3,τ) to ensure that the adjoint preconditioner is equivalent to the transpose of the preconditioner for the primal problem. We note that the formation of the scalar-problems, the transformations to/from characteristics variables, and the application of the inverse of the scalar systems are identical for primal and dual solves only the order of their application are changed. As in the primal case, the preconditioner requires minimal storage, and since the preconditioned operator for the adjoint problem is simply the transpose of operator the primal problem we expect similar performance, enabling us to efficiently compute high-order adjoint solutions.

IV.B. Discrete Space-Time Adjoint System

We now explain how we solve the discrete space-time adjoint system. Consider an unsteady simulation composed of two time slabs. The corresponding space-time discrete residual operator can be compactly expressed as follows:

$$\underbrace{\begin{bmatrix} \mathbf{R}(\cdot) & 0 \\ -\mathbf{G}(\cdot) & \mathbf{R}(\cdot) \end{bmatrix} \begin{pmatrix} \mathbf{V}^n \\ \mathbf{V}^{n+1} \end{pmatrix}}_{\mathcal{R}(\mathcal{V})} = \begin{bmatrix} \mathbf{G}(\mathbf{V}^{n-1}) \\ 0 \end{bmatrix}, \tag{19}$$

where the superscripts denote the time-slabs and $\mathbf{V}$ are the discrete state unknowns which represent the entropy variables in space and time according to Eqn. 9. $\mathbf{R}(\cdot)$ and $\mathbf{G}(\cdot)$ are the discrete nonlinear operators respectively corresponding to the left and righthand side of Eq. 8. It is convenient to solve the nonlinear system in Eqn. 19 by marching forward starting from a state $\mathbf{V}^{n-1}$.

The adjoint system for Eqn. 19 is written as:

$$\begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \mathbf{V}}^T\Big|_{\mathbf{V}^n} & -\frac{\partial \mathbf{G}}{\partial \mathbf{V}}^T\Big|_{\mathbf{V}^{n+1}} \\ 0 & \frac{\partial \mathbf{R}}{\partial \mathbf{V}}^T\Big|_{\mathbf{V}^{n+1}} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\psi}^n \\ \boldsymbol{\psi}^{n+1} \end{bmatrix} = -\frac{\partial \mathcal{J}}{\partial \mathcal{V}} \left(\begin{bmatrix} \mathbf{V}^n \\ \mathbf{V}^{n+1} \end{bmatrix} \right) + \begin{bmatrix} \mathbf{0} \\ \frac{\partial \mathbf{G}}{\partial \mathbf{V}}^T\Big|_{\mathbf{V}^{n+2}} \boldsymbol{\psi}^{n+2} \end{bmatrix}, \tag{20}$$

where the righthand side represents the output linearization with respect to the full spacetime state. Note that it is convenient, but not strictly necessary, to solve Eqn. 20 backwards starting from the adjoint state for the last slab ($\boldsymbol{\psi}^{n+1}$) with an adjoint final condition $\boldsymbol{\psi}^{n+2} = 0$. Analogously to the primal problem, the adjoint of the temporal flux ($\frac{\partial \mathbf{G}}{\partial \mathbf{V}}^T$) couples the adjoint states between time slabs.

In each step of the adjoint solve, the corresponding primal states are required in order to compute the linearizations of $\mathbf{R}$ and $\mathbf{G}$. In this work, we store the primal states for each timeslab in the file-system during the forward solve and read them when computing the adjoint. In a parallel computing environment, writing files to a network file system is not efficient and storing the states can take as long as, or longer than, solving for them. Hence, it is important to reduce the number of file writes via checkpointing and/or to have processes writing the previous slab to file while others are solving for the next slab. Currently, we do not solve and write concurrently as the focus of this paper is the implementation and verification of the matrix-free adjoint infrastructure. Implementation of concurrent solve-and-write and investigation of adaptive checkpointing strategies[30] are ongoing work.

IV.C. Adjoint Operator with Collocated Quadrature

As noted above, a globalized quasi-Newton approach is used to solve the primal equations, where the approximation to Newton's method comes in the use of a collocated quadrature in the Fréchet derivative operator. The motivation for such an approximation is because the collocated quadrature has f_q^4 fewer points, where f_q is the ratio of quadrature-to-nodal points of the de-aliased rule in each of the four directions – three spatial and one temporal. Here, $f_q = 2$, hence the potential speedup is 16. The actual nonlinear speedup is less than that because the slower rate of convergence when the linearization is only approximate. In this paper, we seek to obtain a speedup similar to the nonlinear case but for the adjoint problem which can me cast as a linear system of the form:

$$\mathbf{Q}(x) = \mathbf{A} \cdot x - b = 0, \tag{21}$$

where x is the adjoint state, $\mathbf{A}$ corresponds to the adjoint operator with de-aliased quadrature and b represents the adjoint righthand side computed with the same quadrature as $\mathbf{A}$. The quasi-Newton procedure for Eqn.

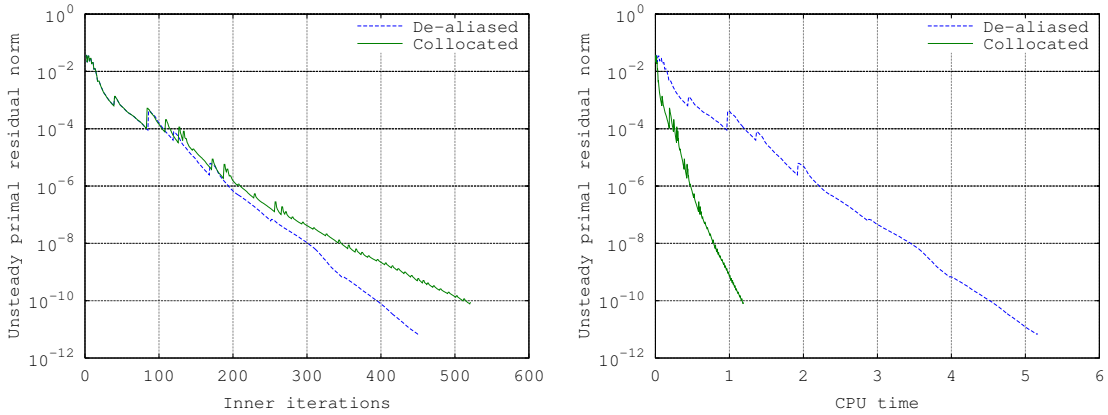
21 is then:

$$\tilde{\mathbf{A}} \cdot (x_{k+1} - x_k) = -\mathbf{Q}(x_k), \tag{22}$$

where $\tilde{\mathbf{A}}$ represents the adjoint operator with the collocated quadrature. We solve Eqn. 22 for the update, $\Delta x_k = (x_{k+1} - x_k)$, using GMRES starting from a guess $x_0 = 0$. Assuming that we solve Eqn. 22 exactly at each GMRES call, the adjoint residual norm behaves as,

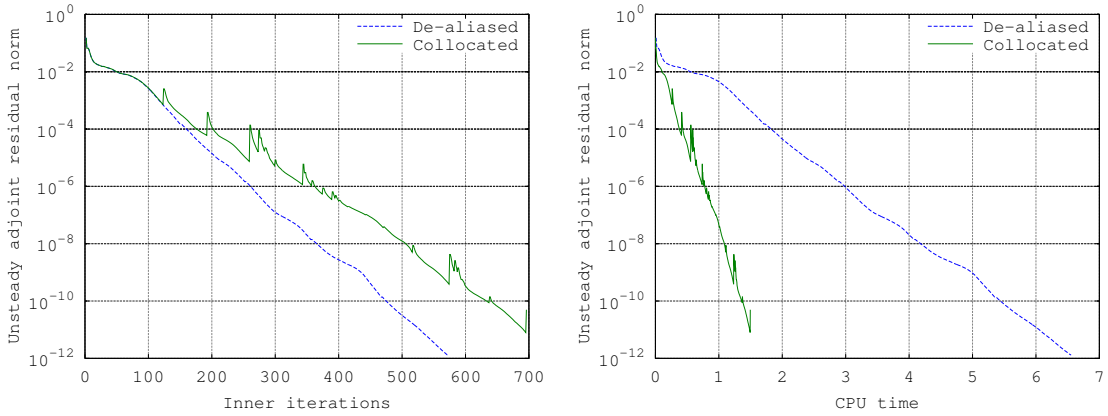
$$||\mathbf{Q}(x_k)|| = ||(\mathbf{I} - \mathbf{A} \cdot \tilde{\mathbf{A}}^{-1})^k \cdot b||, \tag{23}$$

where $\mathbf{I}$ is the identity matrix. Naturally, this procedure can diverge if $\tilde{\mathbf{A}}$ is a poor approximation to $\mathbf{A}$. However, from our experience with the primal problem,[19] we found that the use of a collocated quadrature for the residual Jacobian works well and it indeed offers an overall speed-up over the use of a de-aliased quadrature. The case chosen to quantify the speedup is Window 1 shown in Figure 6(a), discussed later in the text.



(a) Primal residual history versus GMRES iterations.

(b) Primal residual history versus time.



(c) Adjoint residual history versus GMRES iterations.

(d) Adjoint residual history versus time.

Figure 2: Representative residual histories for collocated and de-aliased quadrature in the Fréchet derivative.

Figure 2 shows the primal and adjoint residual convergence for a representative space-time slab for the de-aliased and collocated quadratures. The spikes in the primal residual plots (Figures 2(a) and 2(b)) are due to the nonlinearity of the problem, that is, between outer iterations the residual get reset to the actual nonlinear residual. As expected, in the adjoint case (Figures 2(c) and 2(d)), these spikes are only observed in the collocation curve as the transpose Fréchet derivative is approximate.

Note that collocation in the Fréchet derivative takes roughly 20% more GMRES iterations but, because these iterations are much faster, the overall speedup is approximately a factor of 5 (Figure 2(d)). This speedup is roughly the same for both primal and dual solves. Surprisingly, the adjoint solve is $\sim 30\%$ more

expensive than the primal (Table 1) for both collocation and de-aliasing, in spite of the adjoint problem being linear. Part of the reason for this difference is explained by Figure 1 where the transpose is $\sim 20\%$ more expensive than the forward Fréchet derivative. The remainder of the reason for the difference is unclear and we intend to investigate this further.

Table 1: Speed-up of Fréchet derivative quadratures for case in Section V.B: NACA0012, $M_\infty = 0.5$, $Re = 5 \times 10^3$, $\alpha = 10.0°$, Window 1 (60 cores of Intel Xeon E5-2680v2).

Quadrature	Normalized Total Primal Solve Time	Normalized Total Adj. Solve Time
De-aliased	1.00 (82.59 sec.)	1.00 (107.62 sec.)
Collocated	0.21	0.22

V. Results

We consider two flow conditions over the NACA0012 airfoil, one steady and one unsteady. Although the flows are two-dimensional, we compute them in a three-dimensional, cylindrical domain, as shown in Figure 3, where the two domain faces with z-normals are periodic. The mesh (Fig. 3) consists of 128 elements constructed via a tensor-product of 7^{th}-order polynomials approximating the geometry in each direction. For both flows, the primal and adjoint residuals tolerances are 10^{-10}. The reference length is $C_{\text{ref}} = 1$ and velocity scale is the freestream Mach number, M_∞.

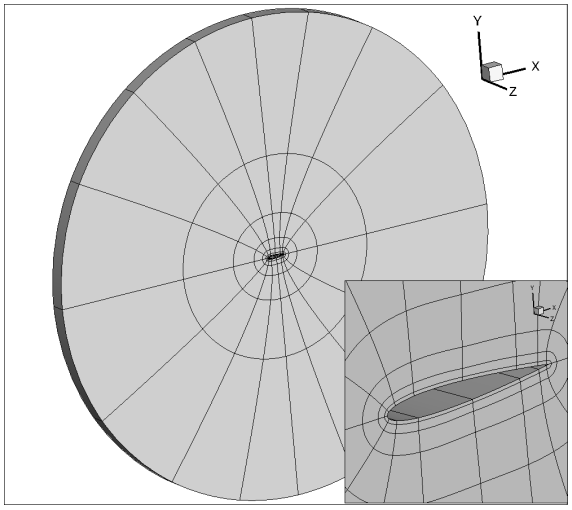


Figure 3: Mesh used for the steady and unsteady runs.

V.A. Steady Viscous Flow

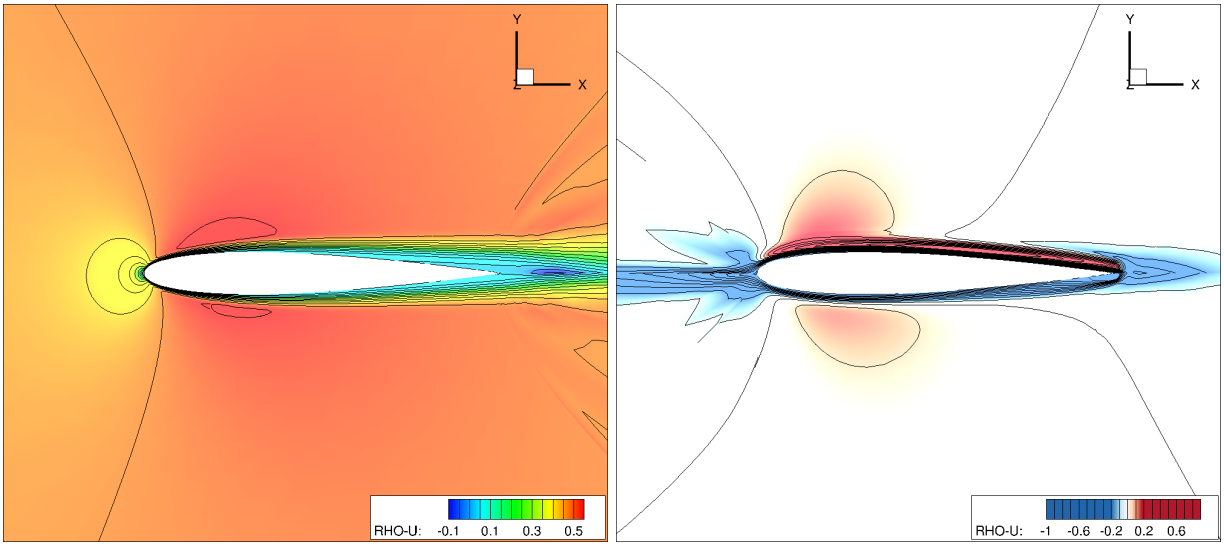
The first case we consider is steady flow at $M_\infty = 0.5$, $\alpha = 1°$ and $Re = 5000$. We compute the primal and adjoint solutions with 2^{nd}, 4^{th}, and 6^{th} spatial approximation orders. Our output of interest is the force acting on the airfoil projected in the x-direction. We use the sensitivity equation in variational form to verify if the adjoint-weighted residual yields the true output variation for small parameter perturbations. The second column of Table 2 lists the "true" output perturbation computed as the difference of the output computed with a perturbed solution, $\mathcal{V}(\alpha+\delta\alpha) = \mathcal{V} \mid \mathcal{R}(\mathcal{V}, \alpha+\delta\alpha) = 0$, and a baseline solution, $\mathcal{V}(\alpha) = \mathcal{V} \mid \mathcal{R}(\mathcal{V}, \alpha) = 0$. The third column of Table 2 lists the adjoint-weigthed residual computed via a discrete inner product between the adjoint solution, $\mathbf{\Psi}$, and the perturbed residual operator, both evaluated with the baseline solution. Note that adjoint-weighted perturbation is only a linear approximation to the true output perturbation and the two columns should only perfectly agree for an infinitesimal parameter perturbation. Nonetheless, the fourth column verifies that the output perturbation computed via the adjoint-weighted residual agrees well with

the direct output difference. Note that we perturb the angle of attack by 1% ($\delta\alpha = 0.01°$) and the error in the computed perturbations is within 1%.

Table 2: Steady adjoint sensitivity verification for NACA0012, $M_\infty = 0.5$, $Re = 5 \times 10^3$, $\alpha = 1.0°$, and $\delta\alpha = 0.01°$.

Approx. Order	$\delta\mathcal{J} = \mathcal{J}(\mathcal{V}(\alpha+\delta\alpha)) - \mathcal{J}(\mathcal{V}(\alpha))$	$\delta\mathcal{J}_{\text{adj}} = \boldsymbol{\Psi}^T\mathcal{R}(\mathcal{V}(\alpha), \alpha+\delta\alpha)$	$(\delta\mathcal{J} - \delta\mathcal{J}_{\text{adj}})/\delta\mathcal{J}$
2	$-2.2392e-06$	$-2.2283e-06$	0.49%
4	$-4.3091e-07$	$-4.2989e-07$	0.24%
6	$5.2794e-07$	$5.2622e-07$	0.33%

Figure 4 shows the 6$^{\text{th}}$-order primal and x-force-adjoint fields for x-momentum. The non-white regions in Figure 4(b) are regions where residuals in the x-momentum conservation equation will affect the x-force. These results are similar to adjoint results obtained previously by other researchers.



(a) 6$^{\text{th}}$-order primal x-momentum.

(b) 6$^{\text{th}}$-order x-force-adjoint x-momentum.

Figure 4: NACA0012, $M_\infty = 0.5$, $Re = 5 \times 10^3$, $\alpha = 1.0°$: primal and x-force adjoint solutions.

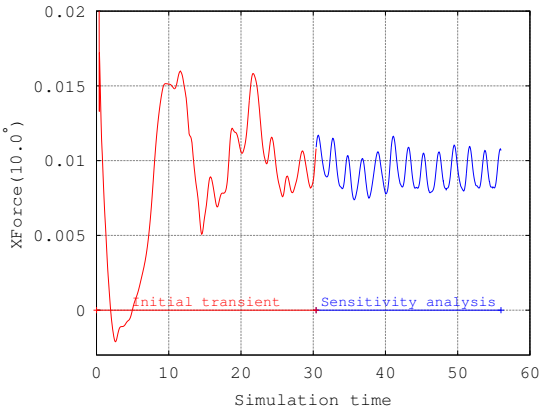
V.B. Unsteady Viscous Flow

The second case we consider is unsteady flow at $M_\infty = 0.5$, $Re = 5 \times 10^3$ and $\alpha = 10.0°$ with 4$^{\text{th}}$-order approximation in space and time. We initialize the flow with freestream conditions and it undergoes an initial transient marked in red in Figure 5(a). We use the flow state (Figure 7(a)) at the end of this period as the initial condition for the sensitivity analyses.

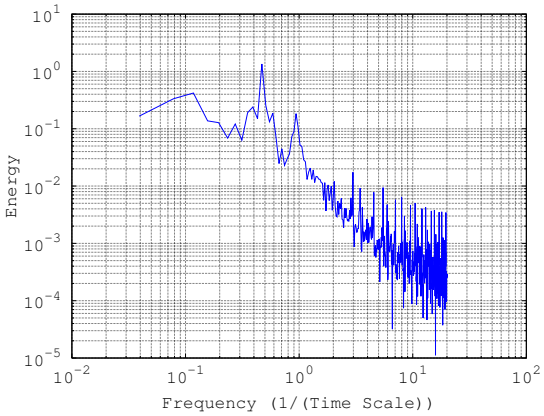
After the initial transient, the flow is expected to exhibit regular vortex sheding at a frequency (f) corresponding to Strouhal number of $St = \frac{fL}{|\boldsymbol{u}_\infty|} \sim 0.2$ based on the body's cross-section length scale (L). In fact, the Strouhal number computed with the peak frequency shown in Figure 5(b) is $St = 0.163$ using a reference length, $L = C_{\text{ref}} \sin(\alpha)$.

The output of interest for this case is the integral over time of the x-force. We consider four time window sizes (Figure 6(a)) for the force integration. Note that the average x-force for windows 3 and 4 are within 1% of each other. Figure 6(b) shows the force difference in between the perturbed solution and the baseline solution for Window 4.

Krakos[31] Heaviside-weighted integration windows introduce error when one is interested in computing the infinite-time average of an outputs. We emphasize that the purpose of this study is only to assess the correctness of the adjoint sensitivities, hence Heaviside-weighted integration windows are adequate. If we



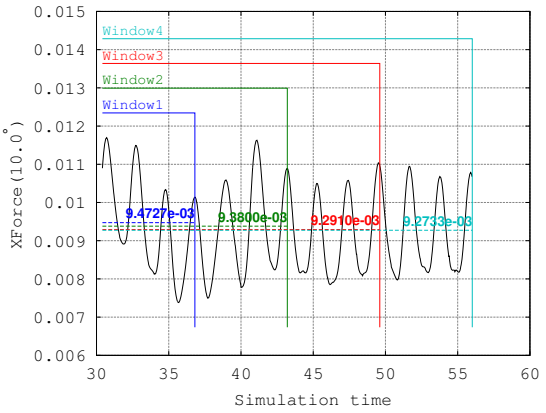
(a) Complete time history of x-force signal.



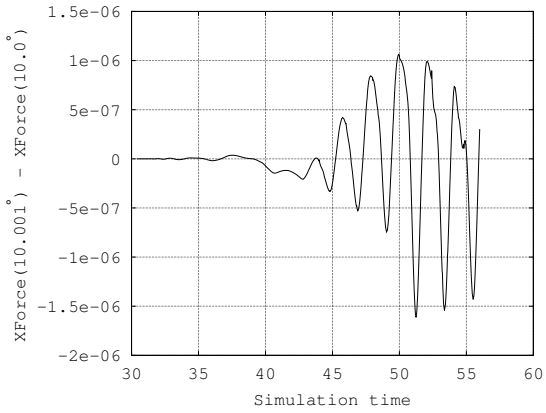
(b) Spectrum of x-force in the sensitivity analysis portion of the signal.

Figure 5: Time history and energy spectrum of the x-force signal.

were interested in the true sensitivity of the mean x-force for infinite time, smooth weighting functions should be used with finite windows.



(a) Windows for x-force time integration.



(b) Time history for the difference in x-force for Window 4.

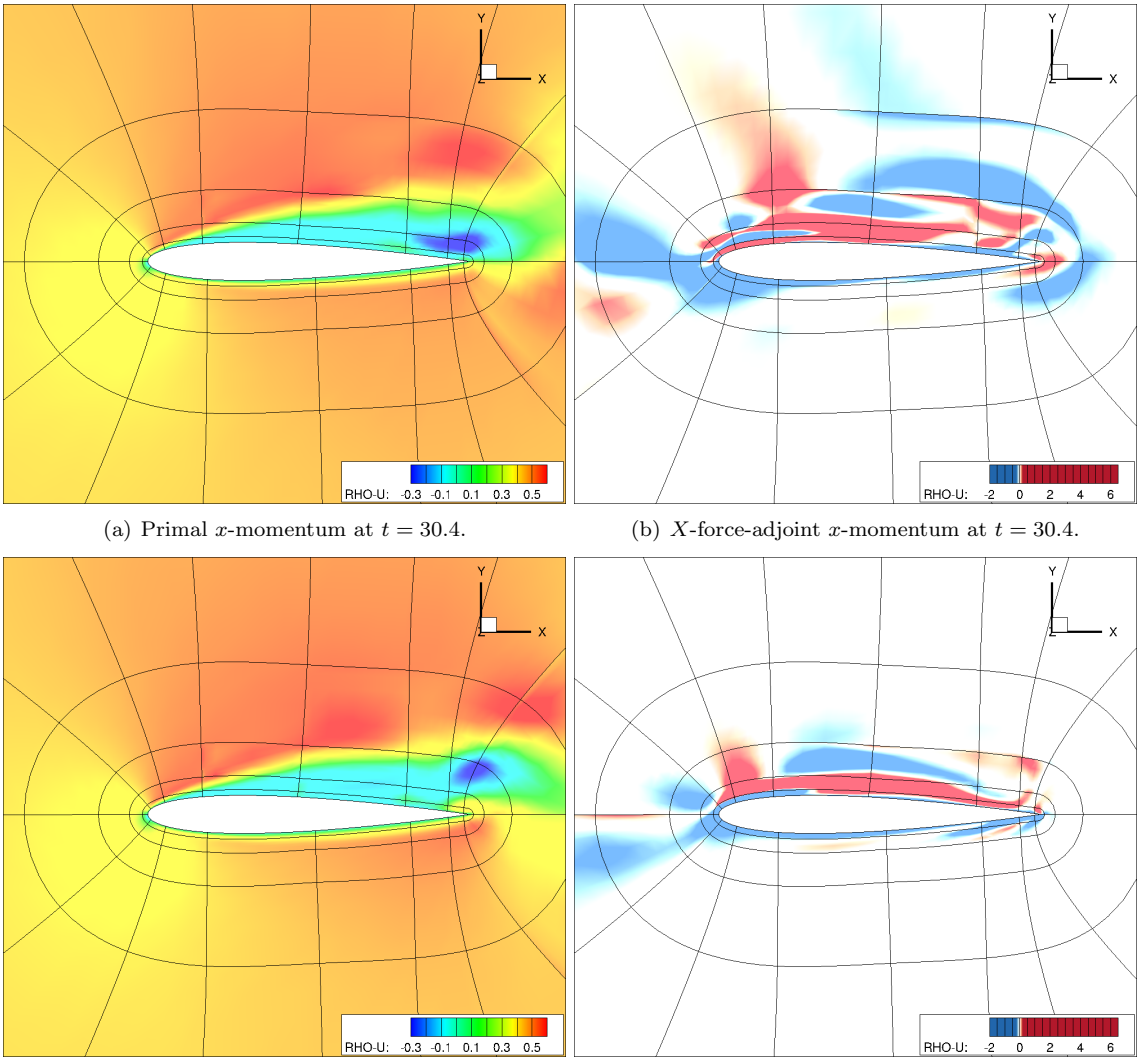
Figure 6: Time history of the x-force in the sensitivity analysis time frame.

Similarly to the steady case, we compute the output for each of the time windows with the baseline solution and their corresponding adjoints. Then we compute the output perturbation via the adjoint-weighted residual and via the direct difference from the perturbed solution. The results of the sensitivity analyses are shown in Table 3. Note that the computed perturbations are within 1% of each other for all windows.

Table 3: Unsteady adjoint sensitivity verification for NACA0012, $M_\infty = 0.5$, $Re = 5 \times 10^3$, $\alpha = 10.0°$.

Window width	$\delta\mathcal{J} = \mathcal{J}(\mathcal{V}(\alpha+\delta\alpha)) - \mathcal{J}(\mathcal{V}(\alpha))$	$\delta\mathcal{J}_{\text{adj}} = \mathbf{\Psi}^T\mathcal{R}(\mathcal{V}(\alpha), \alpha+\delta\alpha)$	$\delta\alpha$	$(\delta\mathcal{J} - \delta\mathcal{J}_{\text{adj}})/\delta\mathcal{J}$
6.40	$-7.90941e-08$	$-7.93761e-08$	$0.010°$	-0.36%
12.80	$-2.21309e-06$	$-2.21259e-06$	$0.005°$	0.02%
19.20	$-2.06548e-07$	$-2.05836e-07$	$0.001°$	0.35%
25.60	$-3.67374e-07$	$-3.65131e-07$	$0.001°$	0.61%

Figures 7(b) and 7(d) show the x-momentum adjoint solution for Window 4 at two times instances. The non-white regions of the domain are areas where we need to approximate well the primal solution so we obtain an accurate output calculation. The complexity of the adjoint field emphasizes the importance of adaptive methods in unsteady flow problems.



(a) Primal x-momentum at $t = 30.4$.

(b) X-force-adjoint x-momentum at $t = 30.4$.

(c) Primal x-momentum at $t = 43.6$.

(d) X-force-adjoint x-momentum at $t = 43.6$.

Figure 7: 4^{th}-order primal and x-force adjoint solution for Window 4.

VI. Conclusions

Our end goal with adjoints is to adaptively allocate degrees of freedom in space and time to efficiently and accurately simulate separated flows. Here, we presented the first step towards that goal: the implementation of an efficient adjoint solver. The novelty of this work is the ability to efficiently compute space-time adjoints for high orders of approximation accuracy. This was achieved by a matrix-free approach which allows us to scale up the approximation order in both space and time.

For the transpose Fréchet derivative, we use most of the framework layed out for the forward problem including the matrix-free ADI preconditioner which is simply transposed to be applied in the adjoint problem. This procedure works well for the problems tested. Furthermore, the quasi-Newton aproach for the adjoint problem presented here provides virtually the same speed up as a similar procedure for the forward problem.

The next steps for our framework are to implement a procedure for concurrenty writing the primal and

adjoint states' slabs while solving, investigate the reason why the adjoint problem is more expensive than the primal in the unsteady case, and to develop an efficient output-based adaptive procedure for unsteady problems.

Finally, as we move on to complex applications, we expect to encounter chaotic problems where the adjoint procedure as-is breaks down. Nevertheless, the framework presented here is a building block for approaches like the Least-Squares Shadowing of Wang *et al.*[32]

References

[1]Leschziner, M., "Modelling turbulent separated flow in the context of aerodynamic applications," *Fluid Dynamics Research*, Vol. 38, No. 2-3, 2006, pp. 174–210.

[2]Levy, D. W., Laflin, K. R., Tinoco, E. N., Vassberg, J. C., Mani, M., Rider, B., Rumsey, C. L., Wahls, R. A., Morrison, J. H., Brodersen, O. P., Crippa, S., Mavriplis, D. J., and Murayama, M., "Summary of Data from the Fifth Computational Fluid Dynamics Drag Prediction Workshop," *Journal of Aircraft*, 2014.

[3]Slotnik, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D., "CFD Vision 2030 Study: A Path to Revolutionary Computational Sciences," Technical Report 20140003093, NASA, 2014.

[4]Fidkowski, K. J. and Darmofal, D. L., "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics," *AIAA Journal*, Vol. 49, No. 4, 2011, pp. 673–694.

[5]Bey, K. S., *An hp-adaptive discontinuous Galerkin method for hyperbolic conservation laws*, PhD dissertation, University of Texas at Austin, 1994.

[6]Heuveline, V. and Rannacher, R., "Duality-based Adaptivity in the *hp*-finite element method," *Journal of Numerical Mathematics*, Vol. 11, No. 2, 2003, pp. 95–113.

[7]Rachowicz, W., Pardo, D., and Demkowicz, L., "Fully Automatic *hp*-Adaptivity in Three Dimensions," Tech. Rep. 04-22, ICES, 2004.

[8]Houston, P. and Süli, E., "A note on the design of *hp*-adaptive finite element methods for elliptic partial differential equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, 2005, pp. 229–243.

[9]Giani, S. and Houston, P., "High–Order *hp*–Adaptive Discontinuous Galerkin Finite Element Methods for Compressible Fluid Flows," *ADIGMA - A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications*, edited by N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. van der Ven, and K. Sørensen, Vol. 113 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, Springer Berlin / Heidelberg, 2010, pp. 399–411.

[10]Burgess, N. K. and Mavriplis, D. J., "An *hp*-Adaptive Discontinuous Galerkin Solver for Aerodynamic Flows on Mixed-Element Meshes," *49th AIAA Aerospace Sciences Meeting and Exhibit*, No. AIAA 2011-490, 2011.

[11]Ceze, M. and Fidkowski, K. J., "Anisotropic *hp*-adaptation framework for functional prediction," *AIAA Journal*, Vol. 51, No. 2, February 2013, pp. 492–509.

[12]Park, M. A., *Anisotropic Output-Based Adaptation with Tetrahedral Cut Cells for Compressible Flows*, PhD dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.

[13]Loseille, A. and Löhner, R., "Anisotropic Adaptive Simulations in Aerodynamics," *48th AIAA Aerospace Sciences Meeting and Exhibit*, No. AIAA 2010-169, 2010.

[14]Yano, M., *An Optimization Framework for Adaptive Higher-Order Discretizations of Partial Differential Equations on Anisotropic Simplex Meshes*, PhD dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 2012.

[15]Barth, T., *Space-time error representation and estimation in Navier-Stokes calculations*, Vol. 26 of *Lecture Notes in Computational Science and Engineering*, Springer Berlin Heidelberg, 2007, pp. 29–48.

[16]Taube, A., Gassner, G., and Munz, C.-D., "HP-Adaptation in Space-Time within an Explicit Discontinuous Galerkin Framework," *ADIGMA - A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications*, edited by N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. van der Ven, and K. Sørensen, Vol. 113, Springer Berlin Heidelberg, 2010, pp. 427–439.

[17]Fidkowski, K. J. and Luo, Y., "Output-Based Space-Time Mesh Adaptation for the Compressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 230, 2011, pp. 5753–5773.

[18]Diosady, L. T. and Murman, S. M., "Design of a Variational Multiscale Method for Turbulent Compressible Flows," *21th AIAA Computaional Fluid Dynamics Conference*, 2013.

[19]Diosady, L. T. and Murman, S. M., "DNS of flows over periodic hills using a discontinuous Galerkin spectral element method," AIAA paper, 2014, accepted for publication.

[20]Diosady, L. T. and Murman, S. M., "Higher-Order Methods for Compressible Turbulent Flows Using Entropy Variables," *53rd AIAA Aerospace Sciences Meeting and Exhibit*, 2015.

[21]Hughes, T. J. R., Franca, L., and Mallet, M., "A new finite element formulation for computational fluid dynamics: I Symmetric forms of the compressible Euler and Navier-Stokes equations and the second law of thermodynamics," Vol. 54, 1986, pp. 223–234.

[22]Ismail, F. and Roe, P. L., "Affordable, Entropy-consistent Euler flux functions II: entropy production at shocks," *J. Comput. Phys.*, Vol. 228, No. 15, Aug. 2009, pp. 5410–5436.

[23]Kirby, R. M. and Karniadakis, G. E., "De-aliasing on non-uniform grids: algorithms and applications," *Journal of Computational Physics*, Vol. 191, 2003, pp. 249–264.

[24]Diosady, L. T. and Murman, S. M., "Design of a variational multiscale method for turbulent compressible flows," AIAA Paper 2013-2870, 2013.

[25]Vos, P., Sherwin, S., and Kirby, R., "From h to p Efficiently: Implementing finite and spectral/hp element discretizations to achieve optimal performance at low and high order approximations." *Journal of Computational Physics*, Vol. 229, No. 13, 2010, pp. 5161–5181.

[26]Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856–869.

[27]Knoll, D. A. and Keyes, D. E., "Jacobian-free Newton-Krylov methods: a survey of approaches and applications," *Journal of Computational Physics*, Vol. 193, No. 1, 2004, pp. 357–397.

[28]Diosady, L.T. and Murman, S.M., "Tensor-Product Preconditioners for Higher-Order Space-Time Discontinuous Galerkin Methods," *Journal of Computational Physics*, Vol. submitted, 2015.

[29]Ceze, M. and Murman, S.M., "Global Convergence Strategies for a Spectral-Element Space-Time Discontinuous-Galerkin Discretization of the Navier-Stokes Equations," *27th International Conference on Parallel Computational Fluid Dynamics*, 2015.

[30]Wang, Q., Moin, P., and Iaccarino, G., "Minimal Repetition Dynamic Checkpointing Algorithm for Unsteady Adjoint Calculation," *SIAM Journal on Scientific Computing*, Vol. 31, No. 4, 2009, pp. 2549–2567.

[31]Krakos, J., *Unsteady Adjoint Analysis for Output Sensitivity and Mesh Adaptation*, Ph.D. thesis, Massachusetts Institute of Technology, 2012.

[32]Wang, Q., Hu, R., and Blonigan, P., "Least Squares Shadowing sensitivity analysis of chaotic limit cycle oscillations," *Journal of Computational Physics*, Vol. 267, No. 0, 2014, pp. 210 – 224.